TABLE 3

| NAME | DESCRIPTION |
|---|---|
| DATA | "Data Port". It is used as data port and have the same definition in ATA specification. |
| LENGTH | "Transfer Length". The register specifies the transfer byte count in Hardware cycles. |
| CTL | "Control" mapped to Flash ROM pins of cs#, wr#, and oe#. Host can use the register to control the status of the pins. |
| DBUS | "Data Bus" mapped to Flash ROM pins of data bus. Host can use the register to control the status of the pins. |
| ABUSLOW | "Address Bus Low" mapped to Flash ROM pins of address bus low byte. Host can use the register to control the status of the pins. |
| ABUSHIGH | "Address Bus High" mapped to Flash ROM pins of address bus high byte. Host can use the register to control the status of the pins. |
| DRIVE SELECT | "Drive Select". Host writes the register to select IDE devices. It has the same definition in ATA specification. |
| COMMAND/ STATUS | "Command/Status". Host writes the register to issue an ATA command and reads the register to obtain the device status. It has the same definition in ATA specification. |

In the manner of the invention, the flash ROM **202** can be directly programmed to update, for example, a firmware code through the IDE interface by simply redefining the registers of the ATA task files used by the IDE interface. There is no need to go through a route inside the periphery device. The programming process can be achieved by a software method or a hardware method, or even by a mixed method of those two. The software method includes one step that the host directly reads or writes the flash ROM through the redefined ATA task files.

The hardware method includes two steps that data transferred between the HOST and the flash ROM are first stored into a buffer, such as a RAM, and then the data on the RAM are written into the flash ROM for a write cycle, or are read by the Host. The IDE interface is released for other uses after data are temporarily stored into the RAM.

The invention has been described using an exemplary preferred embodiment. However, it is to be understood that the scope of the invention is not limited to the disclosed embodiment. On the contrary, it is intended to cover various modifications and similar arrangements. The scope of the claims, therefore, should be accorded the broadest interpretation so as to encompass all such modifications and similar arrangements.

What is claimed is:

**1**. A method for updating a firmware of a flash read-only memory (ROM), the method being suitable for a host computer (Host) to update the firmware of the flash ROM through an integrated device electronics (IDE) interface, the method comprising:

re-interpreting all IDE bus activities so that the flash ROM is updated without using definitions of the IDE interface;

redefining registers of task files from definitions of the IDE interface to definitions for updating the flash ROM;

entering a flash ROM programming mode by issuing an IDE control command from the Host, and the registers of the task files work under the definitions for updating the flash ROM;

reading or writing data on the flash ROM by the Host using the redefined task files of the IDE interface through a software cycle or a hardware cycle, in which the flash ROM has a plurality of pins;

leaving the flash ROM programming mode and returning to a normal mode with original definitions of the task files.

**2**.The method of claim **1**, wherein the step of reading or writing data on the flash ROM further comprises assigning an initial address.

**3**. The method of claim **1**, wherein in the step of reading or writing data on the flash ROM, the software cycle comprises directly accessing the pins of the flash ROM by the Host.

**4**. The method of claim **1**, wherein in the step of reading or writing data on the flash ROM, the hardware cycle comprises temporarily storing the data into a memory buffer and then transferring the data from the memory buffer to the Host or the flash ROM depending on reading or writing, in which when the data are completely transferred through the IDE interface, the IDE interface is released for other uses.

**5**. The method of claim **1**, wherein the registers of the task files are redefined, as follows:

a DATA register is used as a data port and has a same definition in the task files;

a LENGTH register is used as a transfer length to specify a transferring byte count of the data in the hardware cycle;

a CTL register is defined as a control mapped to the flash ROM pins of cs#, wr#, and oe#, in which the Host can use the CTL register to control or obtain a status of the pins;

a DBUS register is defined as a data bus mapped to the flash ROM pins of data bus, in which the Host can use the DBUS register to control or obtain a status of the pins; an accompanying bit is defined to decide the direction of data bus;

an ABUSLOW register is defined as an address bus low mapped to the flash ROM pins of an address bus low byte, in which the Host can use the ABUSLOW register to control or obtain a status of the pins;

an ABUSHIGH register is defined as an address bus high mapped to the flash ROM pins of an address bus high byte, in which the Host can use the ABUSHIGH register to control or obtain a status of the pins;

a DRIVE-SELECT register is defined as a drive select, in which the Host writes the DRIVE_SELECT register to select an IDE periphery device, and DRIVE_SELECT register has a same definition in the task files; and

a COMMAND/STATUS register with a same definition in the task files, in which the Host writes the

COMMAND/STATUS register to issue an AT attachment (ATA) command, and reads the COMMAND/STATUS register to obtain a programming status.

6. The method of claim 1, wherein the method comprises two commands in the redefined task files used to enter or leave the flash ROM programming mode, and two commands to start reading or writing the data on the flash ROM.

7. The method of claim 1, wherein the original definition of the task files used by the IDE interface is an AT attachment (ATA) specification.

8. The method of claim 1, wherein when the flash ROM is processed, other accesses to the flash ROM are temporarily inhibited.

9. A system for programming a flash read-only memory (ROM), the system communicating with a host computer (Host) through an integrated device electronics (IDE) interface, the system comprising:

a flash controller, which is coupled to the Host through the IDE interface and interprets task files used by the Host to write data into the flash ROM or read data from the flash ROM;

the flash ROM, coupled to the flash controller; and

a microprocessor, coupled to the flash controller, wherein the microprocessor inhibits the access to the flash ROM while the flash ROM is in programming;

wherein when the flash ROM is to be updated, the Host redefines registers of the task files so that a plurality of control commands and the data are transferred between the system and the Host through the IDE interface, and so that original definitions of the IDE interface is changed and re-defined according to the re-defined registers of the task files;

when the Host requests to access the flash ROM, the Host switches the system into a flash ROM programming mode through a flash_on command of the control commands under the re-defined IDE interface and the re-defined registers of the task files, and leaves the flash ROM programming mode through a flash_off command of the control commands to return the original definitions of the IDE interface and the registers of the task files; and

the flash controller receives read/write activities to the redefined task files from the Host and interprets these redefined activities to perform writing or reading the data on the flash ROM through a software cycle or a hardware cycle.

10. The system of claim 9, wherein when the system is operated in the software cycle, the Host can directly control flash ROM pins to transfer the data through the flash controller.

11. The system of claim 10, wherein when the system is operated in the software cycle, some registers of the task files are redefined as follows:

a CTL register is defined as a control mapped to the flash ROM pins of cs#, wr#, and oe#, in which the Host can use the CTL register to control a status of the pins;

a DBUS register is defined as a data bus mapped to the flash ROM pins of data bus, in which the Host can use the DBUS register to control a status of the pins;

an ABUSLOW register is defined as an address bus low mapped to the flash ROM pins of an address bus low byte, in which the Host can use the ABUSLOW register to control a status of the pins; and

an ABUSHIGH register is defined as an address bus high mapped to the flash ROM pins of an address bus high byte, in which the Host can use the ABUSHIGH register to control a status of the pins.

12. The system of claim 9, wherein when the system is operated in the hardware cycle, the data are temporarily stored in a memory buffer, and then the flash controller transfers the data to either the Host or the flash ROM, depending on a desired purpose of writing or reading, in which as the data are completly transferred through the IDE inter-face, the IDE interface is released and returns to its original definition on the task files.

13. The system of claim 12, wherein the memory buffer comprises a random access memory (RAM) coupled to the flash ROM.

14. The system of claim 12, wherein when the system is operated in the hardware cycle, some registers of the task files are redefined as follows:

a DATA register is used as a data port and has a same definition in the task files;

a LENGTH register is used as a transfer length to specify a transferring byte count of the data in the hardware cycle; and

a COMMAND/STATUS register with a same definition in the task files, in which the Host writes the COMMAND/STATUS register to issue an AT attachment (ATA) command, and reads the COMMAND/STATUS register to obtain a programming status.

15. The system of claim 9, wherein the software cycle and the hardware cycle can be independently used or in mixed use so as to achieve an in-system flash ROM programming.

16. The system of claim 9, wherein the flash ROM comprises a pin structure with 64Kx8.

17. The system of claim 9, wherein the original definition of the task files used by the IDE interface is an AT attachment (ATA) specification.

18. The system of claim 9, wherein after redefinition, the task files comprises registers with definition as follows:

a DATA register is used as a data port and has a same definition in the task files;

a LENGTH register is used as a transfer length to specify a transferring byte count of the data in the hardware cycle;

a CTL register is defined as a control mapped to the flash ROM pins of cs#, wr#, and oe#, in which the Host can use the CTL register to control a status of the pins;

a DBUS register is defined as a data bus mapped to the flash ROM pins of data bus, in which the Host can use the DBUS register to control a status of the pins;

an ABUSLOW register is defined as an address bus low mapped to the flash ROM pins of an address bus low byte, in which the Host can use the ABUSLOW register to control a status of the pins;

an ABUSHIGH register is defined as an address bus high mapped to the flash ROM pins of an address bus high byte, in which the Host can use the ABUSHIGH register to control a status of the pins;

a DRIVE_SELECT register is defined as a drive select, in which the Host writes the DRIVE_SELECT register to select an IDE periphery device, and DRIVE_SELECT register has a same definition in the task files; and

a COMMAND/STATUS register with a same definition in the task files, in which the Host writes the COMMAND/STATUS register to issue an AT attachment (ATA) command, and reads the COMMAND/STATUS register to obtain a programming status.

* * * * *